Aggregating Distributed Sensor Data for Network Intrusion Detection

JOHN C. McEACHEN, CHENG KAH WAI, and VONDA L. OLSAVSKY
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California
USA
mceachen@nps.edu

Abstract

Distributed network intrusion detection systems which incorporate tens, hundreds, even thousands, of sensors are becoming increasing popular. Managing and presenting the information from these sensors is becoming an increasingly difficult task. This paper explores the use of Conversation Exchange Dynamics (CED) to integrate and display sensor information from multiple nodes. We present an experimental setup consisting of multiple sensors reporting individual findings to a central server for aggregated analysis. Different scenarios of network attacks and intrusions were planned to investigate the effectiveness of the distributed system. The network attacks were taken from the M.I.T Lincoln Lab 1999 Data Sets. The distributed system was subjected to different combinations of network attacks in various parts of the network. The results were then analyzed to understand the behavior of the distributed system in response to the different attacks. In general, the distributed system detected all attacks under each scenario. Some surprising observations also indicated attack responses occurring in unanticipated scenarios.

1. Introduction

Intrusion Detection Systems (IDS) act as an additional layer of security to the organization's perimeter defense, which usually, is implemented using firewalls. Firewalls are effective in preventing unauthorized entry into the organization's network. However, firewalls cannot detect unauthorized behavior that is present in network traffic they allow to go through.

Understanding network behavior for the purposes of diagnosis and intrusion detection is currently a major effort in the quest to build secure, robust and dependable computing systems. Specifically, intrusion detection systems (IDS) are detection security mechanisms that monitor a computer system or network, attempt to detect malicious activity, and raise an alarm to system or security administrators. IDSs can be classified as either anomaly-based detection or signature-based detection [1]. The former approach detects anomalous behavior, which may be a superset of undesirable behavior, and generally suffers from high false alarm rates. The latter signature-based approach may reduce false alarm rates but generally depends on a well-defined security policy to base detection on. Furthermore, signature-based intrusion detection systems are unable to detect events for which a signature is not defined in their signature database.

Recently, the incidence and scope of network denial of service (DoS) attacks has increased dramatically. The scope of DoS resulting from worm attacks is quick and wide-spread, even global, in nature. Current network security mechanisms, however, insufficiently handle the threat of worm attacks. Moore et al [2] argue that automated containment of self-propagating code is more likely to be successful than prevention or treatment mechanisms. Successful containment mechanisms require rapid and accurate detection of network-based and host-based events such that human operators can efficiently respond to attacks.

Recent work in intrusion detection has moved in this direction. Bro is a real-time network IDS that emphasizes high-speed network monitoring and separation of detection mechanisms from security policy [3]. Efforts to deal with network protocol ambiguities have utilized semantic information. Sommer and Paxson extended Bro with a context-based signature matching capability to improve upon basic signature-matching using regular expressions [4]. Shankar and Paxson extended Bro with an active network topology mapping mechanisms to handle network evasion attacks [5]. Krugel, Toth, and Kirda developed an anomaly detection system to detect malicious network packet payloads by exploiting application level information [6]. Low-rate probing and DoS attacks can be handled in a similar manner [7].

Network-based IDS can be further specified based on their structure. All of the above efforts can be considered centralized IDS. Centralized IDS operate standalone, with centralized applications physically integrated within a box, while distributed IDS consist of multiple IDS over a large network, all of which communicate with each other.

The biggest shortcoming in centralized, standalone IDS is that they are built on a single physical entity, which is responsible for both collecting and analyzing data. This can impose severe limitations on efficiency and the system resources, especially when a high volume of data needs to be processed. Distributed IDS (DIDS) can overcome this shortcoming by performing distributed data collection and possibly preprocessing, depending on the design of the system.

DIDS consist of multiple sensors deployed in different areas of a large network, all of which report to a central server that aggregates the information and processes it. The sensors should ideally be deployed on separate network segments and geographical locations [1].

Several established efforts in DIDS are on-going. Oft cited efforts include [8], [9] and [10]. These efforts tend to focus on the collection and distribution architecture of the DIDS. Unfortunately, how this information is collated, managed and presented is not addressed.

The area of interest in this research is implementing, analyzing and presenting sensor information from a distributed IDS using conversation exchange dynamics (CED) [11]. This is achieved by distributing the sensors on different network segments to monitor the traffic in different parts of the network. The sensors relay the information to a core component, which converts it into a pre-defined format before analyzing it and displaying it on the GUI.

This research is accomplished by analyzing the response of the distributed IDS to network attacks under a variety of conditions. In particular, the Smurf, Mailbomb and Apache2 attacks extracted from the MIT Lincoln Lab IDS datasets will be used to generate a distinctive response [12].

2. Conversation Exchange Dynamics

The underlying concept in [11] is based on the conversation exchange model, which is used to model network traffic. It defines a conversation exchange as an exchange of information between two conversation groups. These conversation groups may represent network nodes, protocols or the tasks which network nodes perform (e.g. client or server). This model uses buckets to represent conversation groups and balls to represent the information that is exchanged between the conversation groups.

As the network moves information around, this is represented as balls being passed between buckets. For example, a series of n packets transmitted from a node N_a to another node N_b would be modeled as n balls moved from bucket *X* and placed in bucket *Y*. The association of node N_a with either bucket *X* or *Y* depends on the nature of the conversation. The bucket can be defined using any combination of conversation characteristics including the affiliation of who is talking (individual hosts or networks), the language they are speaking (TCP, UDP, or ICMP), or the job they are performing (client or server).

In its simplest form, each node in a network is associated with one or more buckets and the total number of packets exchanged between nodes is modeled as moving balls from bucket to bucket. The collection of all buckets together with the allowable distribution range of balls forms a bucket state space.

A bucket b_i is a scalar representing an ordinal number of balls for conversation entity i. As noted, a bucket represents any combination of conversation characteristics, including who is talking (hosts or internal nodes), the language they are speaking (TCP, UDP, or ICMP), or the job they are performing (client or server). A bucket state space of *M* conversation entities is represented as an *M*-dimensional vector space $\vec{b} \equiv (b_1, \ldots, b_M)$. The state vector is a discrete time dependent variable, $\vec{b}_t$; the initial configuration is $\vec{b}_0 \equiv (b_{1,0}, \ldots, b_{M,0})$. The number of possible bucket states, N, can be determined as:

$$N = \binom{M+K-1}{M-1} \tag{1}$$

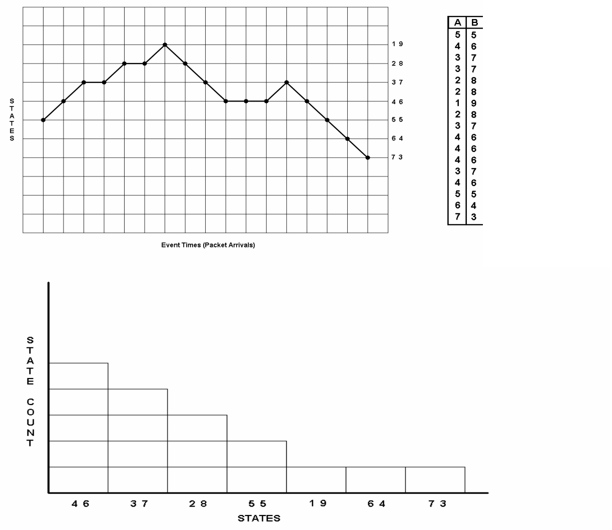
where K is the total number of balls in the system. As an example of the breadth of the bucket space, in our laboratory experiments below, $M = 4$ and $K = 24$, which allows for $N = 118{,}755$ distinct bucket states.

A state transition causes a shift in the distribution of information between the buckets. In other words, this model translates network behavior into bucket state transitions by selecting a ball from the bucket matching the source characteristics of the packet (b_i) and moving that ball into the bucket matching the destination characteristics of the same packet (b_j), thereby redistributing the information and transitioning the state. In a single packet case, this is modeled by changing the bucket state vector $\vec{b}_t \equiv (b_1, \ldots, b_i, \ldots, b_j, \ldots, b_M)$ to $\vec{b}_t \equiv (b_1, \ldots, b_i - 1, \ldots, b_j + 1, \ldots, b_M)$. The number of balls that change buckets in a given time period depends on the number of conversations and the network rate. In general, the net difference in balls between two buckets and the number of buckets for which the net ball count changes is more than one.

Figures 1 and 2 show the importance of the state walk, and how the model provides more information than a simpler model. Both graphs represent a two bucket state space made up of two conversation groups, $\vec{b} \equiv (b_A, b_B)$. The initial state vector is $\vec{b}_0 = (5,5)$. A state transition occurs if a ball is moved from b_A to b_B or vice versa; a removed from b_A and placed back into b_A results in stasis for that time period. Contrast the two bucket and 10 possible states shown in the simple example of figures 1 and 2 with the more realistic bucket state expansion described above.

By examining the manifold, or canyon, developed by collecting various state histograms over time, $\vec{b}_t$, anomalies can be easily spotted as perturbations in the normal flow of the canyon. The cause of such dramatic changes in practice ranges from a single transition to many thousands of packets.

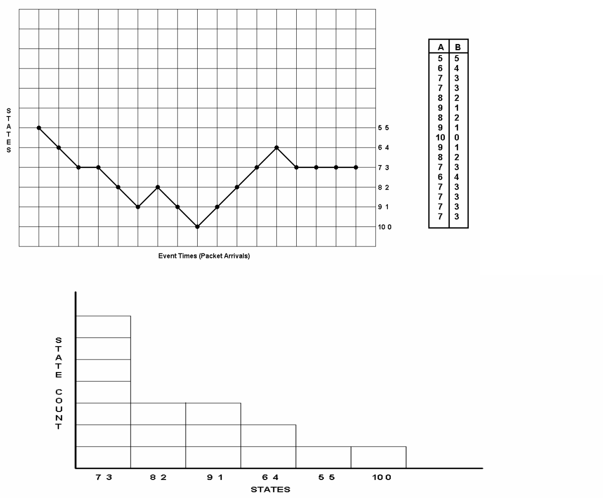
The sequence of *k* state space vectors $\vec{b}_t, \ldots, \vec{b}_{t+k}$ forms a random state space walk. The variable k defines a time window because the system is a discrete random system. The random walks can be reduced into another usable form by ranking states based on state counts, p_i, for a fixed time period. Define $C_{t,t+k}(\vec{b}_i)$ as the number of instances that $\vec{b}_i$ appears in the random walk $\vec{b}_t,\ldots,\vec{b}_{t+k}$. This function defines a histogram $\vec{h} = C_{t,t+k}(\vec{b}_t),\ldots,C_{t,t+k}(\vec{b}_{t+k})$. Rank order h by a permutation $\Pi(h) = p_0,\ldots,p_k$ such that $p_0 \geq p_1 \geq \ldots \geq p_k > 0$.



A	B
5	5
4	6
3	7
3	7
2	8
2	8
1	9
2	8
3	7
4	6
4	6
4	6
3	7
4	6
5	5
6	4
7	3

Fig 1. A sample state walk for a two bucket model. (left) A plot of the state walk over time. (middle) A plot of the visited bucket states, b_i, over the course of the state walk. (Right) A ranked histogram of the bucket states, b_i, over the entire time period of this state walk. Note that not all states have been visited and thus are not included.

An anomaly can cause one of two effects in the above mentioned figures. Either new states are visited, or previously visited states are seen more often. The first effect will cause a spike oriented along the "STATES" axis and the latter along the "STATE COUNT" axis. An anomaly that is orthogonal to the normal traffic flow will tend to cause a spike oriented along the states axis, due to the new states visited. An anomaly that is parallel to the normal traffic flow will tend to cause a spike oriented along the state counts axis, due to the revisiting of previously visited states. The magnitude of the potential spike is what determines the ability of the operator to detect the anomaly. The orientation of the anomaly with respect to the normal traffic flow will determine the magnitude of the perturbation. A single packet anomaly that is orthogonal to the normal traffic flow will cause a large perturbation in the graph (as will be shown later in this paper), where a packet that is parallel to the normal traffic flow will cause a relatively small perturbation. The less orthogonal the anomaly is to the normal traffic flow, the larger the number of anomalous packets required to cause a noticeable perturbation in the graph.



A	B
5	5
6	4
7	3
7	3
8	2
9	1
8	2
9	1
10	0
9	1
8	2
7	3
6	4
7	3
7	3
7	3
7	3

Fig 2. An alternative state walk for a same two bucket model as shown in fig 1. (left) A plot of the state walk over time. (middle) A plot of the bucket sizes over the course of the state walk. (Right) A ranked histogram of the bucket states over the entire time period of this state walk. Note how the histogram varies for this state walk even though both examples end in the same state.

As an example, the buckets in Figure 3 represent eight network nodes. A conversation exchange of *n* network packets between nodes 0 and 2 will result in the movement of *n* balls from bucket 0 to bucket 2. However, the number of balls in each bucket cannot decreased below a minimum level or increased beyond a maximum level, as pre-defined in the decision tree.

The average number of balls in the buckets can be represented in real time on a 3-D graphical display, known as a thermal tower. Information about the network states visited and the number of occurrences can also be accumulated and plotted on a 3-D graphical display, known as a thermal canyon. When there are unusually high counts of certain states, or when there are a large number of states that are usually not visited, it can be an indication of anomalous network activity. Thermodynamic principles of energy, entropy and temperature can be applied to the thermal canyon, which reveal more information about the network health.

Actual implementations using CED often run multiple instantiations of decision trees simultaneous on the same collected data. A decision tree instantiation most often is projection of a certain network policy such as allowing traffic with a certain service to be exchanged with a certain host.

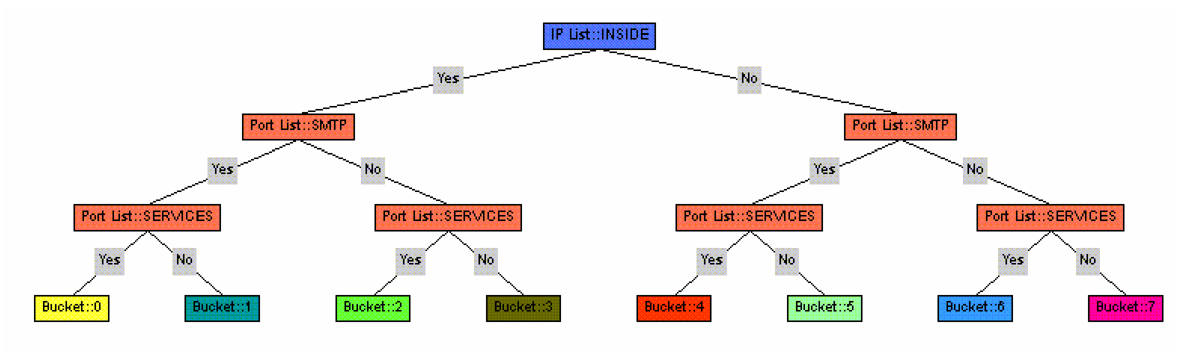


Figure 3. E-mail decision tree instantiation.

Consequently, a typical implementation might include decision trees specifying e-mail, WWW, and FTP instances. Figure 3 is an example of a decision tree instance for e-mail. The aggregating buckets are labeled numerically from left to right, 0 to 7.

Decision tree instances are typically multi-tiered with eight buckets each. In practice, each bucket is initialized with 5 balls, and can have a minimum of 0 balls and a maximum of 10 balls.

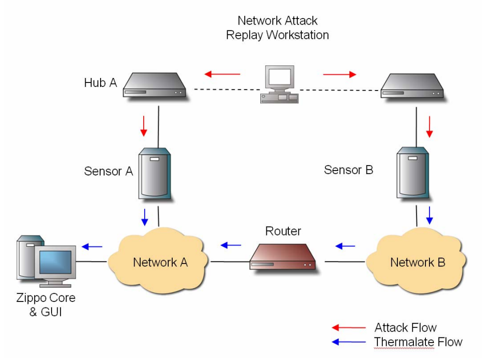


Figure 4. Network topology of experimental setup.

3. Experimental Configuration

Figure 4 shows the network topology of the experiment setup. The network attack replay workstation has two network interface cards, which separately send out pre-recorded network traffic containing both normal user traffic and simulated network attacks. The hubs receive and broadcast the network packets, which are then picked up by sensors A and B.

Sensor A and the CED core component reside on Network A, while sensor B resides on Network B. The two sensors sniff every network packet on their respective network segments and produce sensor-related data – thermalate – that is sent to the CED core. Upon processing and analysis, the core generates thermal canyons that reflect the state of the networks.

In this paper, two general scenarios are tested:

- Sensors A and B detect the same network attack concurrently.
- Sensors A and B detect different network attacks concurrently.

Further, for each scenario we will examine the results for attacks against a service from the perspective of the service's decision tree instantiation and from an orthogonal service's decision tree. These experiments are explained in more detail in the next section.

4. Detecting a Global Attack

The right portion of figure 5 shows the results of the aggregated analysis when both sensors A and B detect the Mailbomb attack at the same time. This is achieved by having the network attack reply workstation replay two instances of 42155148.tcpdump at the same time. The CED core receives two sets of thermalate with approximately the same information.

The left of figure 5 illustrates the thermal canyon of the attack against only a single sensor. Comparing the left of figure 5 with the right, it can be observed the shapes of both canyons are very similar. As might be expected, the difference between the two canyons is that the number of bucket states and the counts for each bucket state have doubled in the right of figure 5. For instance, the highest peak on the canyon floor in the left of figure 5 indicates 80 visited states. The corresponding peak in the right of figure 5 is double that of the left, at approximately 160 states. This behavior was confirmed for all attacks tested.

5. Detecting Simultaneous Local Attacks

In this section, the network attack replay workstation replays the Smurf, Mailbomb and Apache2 attacks on different network segments. The e-mail and WWW decision tree instances are activated on the CED core to perform an aggregated analysis on the thermalate sent from sensors A and B.

5.1 The E-mail Instance

Figure 6 shows the different thermal canyon displays for the e-mail decision tree instance when the Mailbomb attack, the Smurf attack and the combined Mailbomb and Smurf attacks are launched respectively and viewed using the e-mail decision tree instantiation..

We specifically observe that the Smurf attack in the middle of figure 6 shows an orthogonal response to the Mailbomb response in the left of figure 6. Thus in the case of the Smurf attack, even though we do not have an attack targeting the specific service of the decision tree instance, some attack if still observable. Looking closely at the middle of figure 6 during the Smurf attack, the large number of ICMP reply packets from the attackers to the victim result in ball transfers from Bucket 7 to Bucket 3. There are few ball exchanges between the other buckets, which is why there are few bucket states on the thermal canyon.

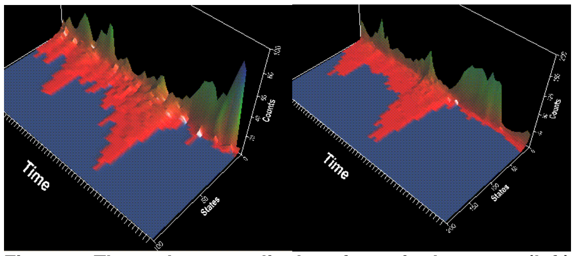


Figure 5. Thermal canyon displays for a single sensor (left) and dual sensors (right) during a Mailbomb attack.

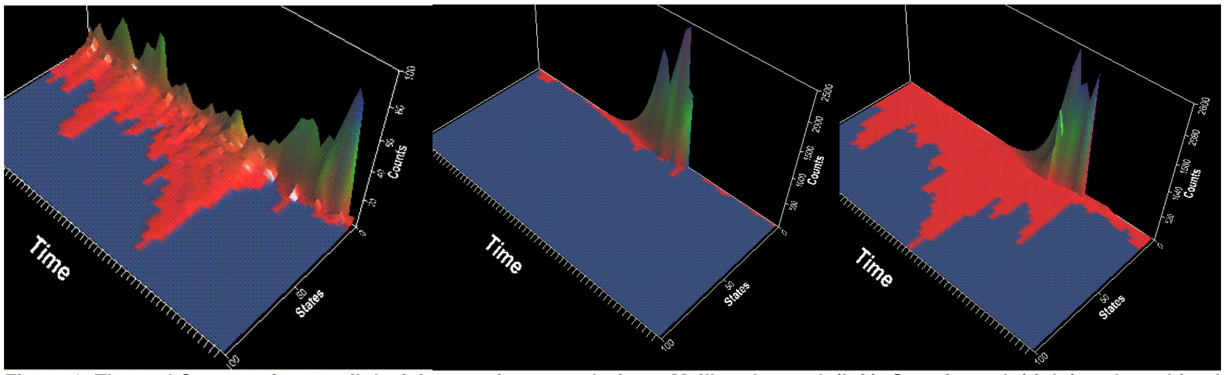


Figure 6. Thermal Canyons for e-mail decision tree instance during a Mailbomb attack (left); Smurf attack (right) and combined Mailbomb and Smurf attacks (right).

5.2 The WWW Instance

Figure 7 shows the different thermal canyons of the WWW decision tree instance, corresponding to the detection of Apache2 attack, Smurf attack and a combination of the two attacks. The bucket definitions for the WWW decision tree instance are listed in table 1.

Again we see an orthogonal response for an attack that is not related to the specific decision tree service instance. In this case, the Smurf attack results in high peaks on the thermal canyon, as shown in the middle of figure 7. The high counts of the states visited are due to massive ball transfers from bucket 7 to 3.

Bucket	Classification
0	N.A
1	Insider IP address with TCP port no. lower than 1024, excluding 80 and 443.
2	Insider IP address with TCP port no. 80 or 443.
3	Insider IP address, no TCP port no. lower than 1024.
4	N.A
5	Outsider IP address with TCP port no. lower than 1024, excluding 80 and 443.
6	Outsider IP address with TCP port no. 80 or 443.
7	Outsider IP address, no TCP port no. lower than 1024.

Table 1. Denotation of Buckets for HTTP PID Instance. Buckets 0 and 4 represent classifications that cannot occur.

5.3 The ICMP Instance

Figure 8 compares the thermal canyon displays of the ICMP decision tree instance to a Smurf attack, an Apache2 attack and combined Smurf and Apache2 attacks respectively. The ICMP decision tree bucket definitions are provide in table 2.

Bucket	Classification
0	Insider IP address with ICMP type 3, 4, 5, 11 or 12.
1	Insider IP address with ICMP type 8 or 17.
2	Insider IP address with ICMP type 0 or 18.
3	Outsider IP address with ICMP type 3, 4, 5, 11 or 12.
4	Outsider IP address with ICMP type 8 or 17.
5	Outsider IP address with ICMP type 0 or 18.
6	Insider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.
7	Outsider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.

Table 2. Denotation of Buckets for ICMP decision tree.

The Apache2 attack results in ball transfers between buckets 6 and 7, as shown in middle of figure 8. The peaks on the thermal canyon clearly indicate an anomalous situation.

During the combined Smurf and Apache2 attacks, the peaks on the thermal canyon are much higher, due to the large combined volume of traffic. Balls are transferred mainly between Buckets 5 and 2, and Buckets 6 and 7. It is difficult to tell from the thermal canyon alone that there are two attacks going on, unless the details about the network packets are obtained from the thermalate.

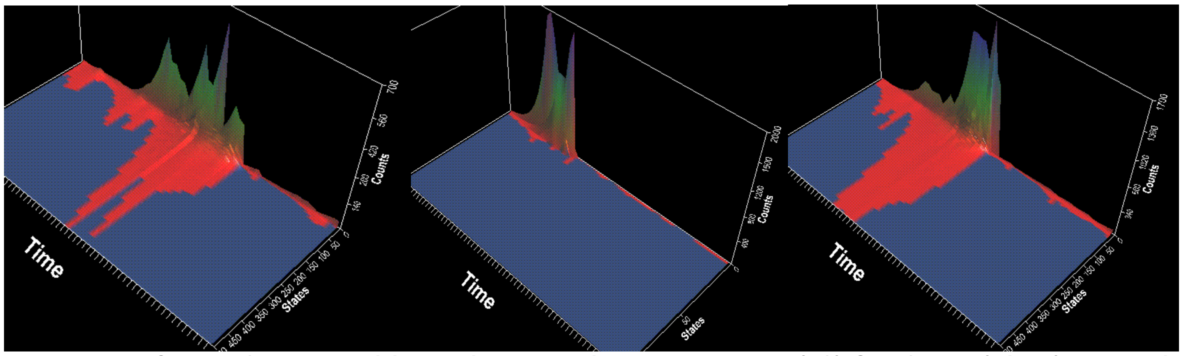


Figure 7. Thermal Canyons for WWW decision tree instance during a Apache2 attack (left); Smurf attack (middle) and combined Apache2 and Smurf attacks (right).

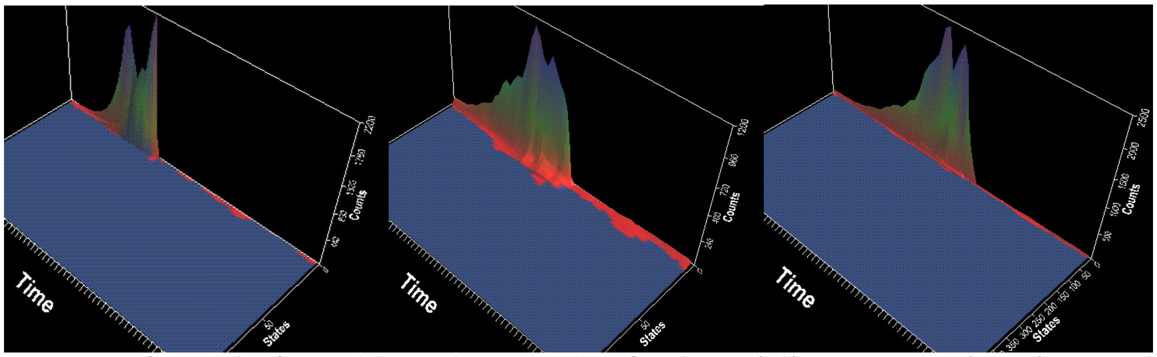


Figure 8. Thermal Canyons for ICMP decision tree instance during a Smurf attack (left); Apache2 attack (middle) and combined Apache2 and Smurf attacks (right).

6. Conclusion

We have presented a novel approach to integrating and presenting aggregated IDS sensor data. We presented several scenarios of attacks directly globally against the network or against particular segments, even simultaneously occurring with attacks on different segments. In each scenario, so form of response was observed, even when the attack was not directed at the particular service of interest. This last point is significant because in intrusion detection, it is more important to have some response that may not exactly match the service under question than to have no response at all. Further, this approach allows us to identify attacks for which no decision tree has been specifically instantiated.

10. References

[1] Axelsson, S. "Intrusion detection systems: A survey and taxonomy", Chalmers University Technical Report 99-15, March 2000.

[2] Moore, D., Shannon, C., Voelker, G. and Savage, S., "Internet quarantine: Requirements for containing self-propagating code," In Proceedings of the IEEE INFOCOM, San Francisco, CA, March 2003.

[3] Paxson, V. "Bro: A system for detecting network intruders in real-time," Computer Networks, 31(23-24), December 1999.

[4] Sommer, R. and Paxson, V. "Enhancing byte-level network intrusion detection signatures with context," In Proceedings of the ACM Conference on Computer and Communications Security (CCS'03), Washington, D.C., November 2003.

[5] Shankar, U. and Paxson, V., "Active mapping: Resisting nids evasion without altering traffic," In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2003.

[6] Krugel, C., Toth, T., and Kirda, E. "Service specific anomaly detection for network intrusion detection," In Proceedings of Symposium on Applied Computing (SAC), Madrid, Spain, March 2002.

[7] Kuzmanovic, A. and Knightly, E., Low-rate tcp-targeted denial of service attacks. In Proceedings of the 2003 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 75–86, Karlsruhe, Germany, 2003.

[8] P. Porras and P. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," *Proceedings of the 20th National Information Systems Security Conference*, Baltimore, MD, October, 1997, pp. 353 – 365.

[9] C. Kruegel, T. Toth, and E. Kirda, "Sparta – A Mobile Agent based Intrusion Detection System." Technical Report, TUV-1841-2002-24, Technical University of Vienna, April 2002.

[10] C. Manikopoulos and S. Papavassiliou, "Network Intrusion and Fault Detection: A Statistical Anomaly Approach," *IEEE Network*, October 2002, pp. 76 – 82.

[11] Donald, S. D., McMillen, R. V., Ford, D. A., and McEachen, J. C., "Therminator 2: A real-time system for patternless intrusion detection", Proc. of the IEEE Military Comms. Conf. (MILCOM 2002), pgs. 1498 – 1502, Los Angeles, October 2002.

[12] Massachusetts Institute of Technology, Lincoln Laboratory, DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/, last accessed October 1, 2005.